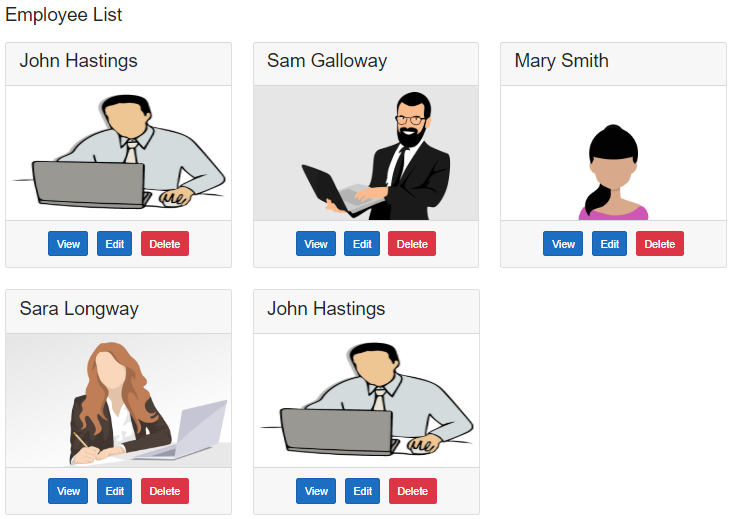
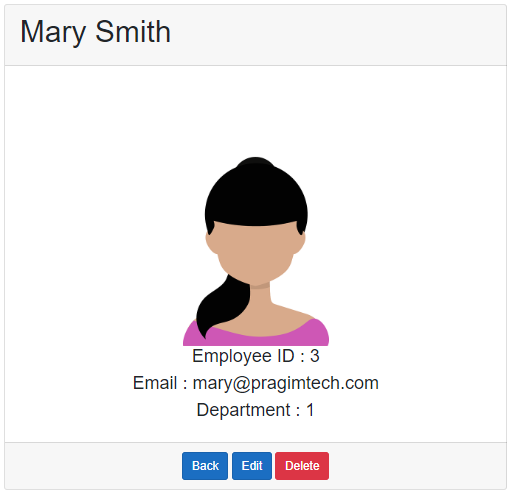
Blazor route parameters

In this video we will discuss route parameters in blazor with an example. The following EmmployeeList component displays the list of all employees.



When we click the View button, the user should be redirected to the URL (/employeedetails/3). The value 3 in the URI is the employee id. The blazor component (EmployeeDetails) displays the details of the Employee.



EmployeeDetails.razor

@page "/employeedetails/{id}"

@inherits EmployeeDetailsBase

<div class="row justify-content-center m-3">

<div class="col-sm-8">

<div class="card">

<div class="card-header">

<h1>@Employee.FirstName @Employee.LastName</h1>

</div>

<div class="card-body text-center">

<img class="card-img-top" src="@Employee.PhotoPath" />

<h4>Employee ID : @Employee.EmployeeId</h4>

<h4>Email : @Employee.Email</h4>

<h4>Department : @Employee.DepartmentId</h4>

</div>

<div class="card-footer text-center">

<a href="/" class="btn btn-primary">Back</a>

<a href="#" class="btn btn-primary">Edit</a>

<a href="#" class="btn btn-danger">Delete</a>

</div>

</div>

</div>

</div>

Copy

To get to this component, employee Id must be passed in the URL as a route parameter. This is specified using the @page directive route template.

@page "/employeedetails/{Id}"

Copy

EmployeeDetailsBase.cs

using EmployeeManagement.Models;

using EmployeeManagement.Web.Services;

using Microsoft.AspNetCore.Components;

using System.Threading.Tasks;

namespace EmployeeManagement.Web.Pages

{

public class EmployeeDetailsBase : ComponentBase

{

public Employee Employee { get; set; } = new Employee();

[Inject]

public IEmployeeService EmployeeService { get; set; }

[Parameter]

public string Id { get; set; }

protected async override Task OnInitializedAsync()

{

Id = Id ?? "1";

Employee = await EmployeeService.GetEmployee(int.Parse(Id));

}

}

}

Copy

* The Id public property is decorated with the [Parameter] attribute.
* The name of the property (Id) matches with the name of the route parameter in the @page directive (@page "/employeedetail/{Id}")
* The blazor router, uses the route parameter value in the URL to populate the Id property in the component class.
* The following 2 conditions must be true, for the router to be able to map the route parameter value in the URL to the public property in the component class
  1. The public property in the component class must be decorated with [Parameter] attribute.
  2. The name of the property in the component class and the name of the route parameter in the @page directive must be the same.

If the names are different you will get the following runtime exception

InvalidOperationException: Object of type 'EmployeeManagement.Web.Pages.EmployeeDetail' does not have a property matching the name 'Id'.

Blazor optional route parameter

Optional parameters aren't supported. However, we can use two @page directives to get the effect of optional parameters.

@page "/employeedetails"

@page "/employeedetails/{Id}"

Copy

The first @page directive allows navigation to the component without a parameter. The second @page directive receives the {Id} route parameter and assigns the value to the Id public property on the component class.

public class EmployeeDetailsBase : ComponentBase

{

public Employee Employee { get; set; } = new Employee();

[Inject]

public IEmployeeService EmployeeService { get; set; }

[Parameter]

public string Id { get; set; }

protected override async Task OnInitializedAsync()

{

// If Id value is not supplied in the URL, use the value 1

Id = Id ?? "1";

Employee = await EmployeeService.GetEmployee(int.Parse(Id));

}

}

Copy

Building URL with route parameter

On the list page when the View button is clicked we want to redirect the user to /employeedetail/id.

<a href="@($"employeedetails/{employee.EmployeeId}")"

class="btn btn-primary m-1">View</a>

Copy

IEmployeeService.cs

public interface IEmployeeService

{

Task<IEnumerable<Employee>> GetEmployees();

Task<Employee> GetEmployee(int id);

}

Copy

EmployeeService.cs

public class EmployeeService : IEmployeeService

{

private readonly HttpClient httpClient;

public EmployeeService(HttpClient httpClient)

{

this.httpClient = httpClient;

}

public async Task<Employee> GetEmployee(int id)

{

return await httpClient.GetJsonAsync<Employee>($"api/employees/{id}");

}

public async Task<IEnumerable<Employee>> GetEmployees()

{

return await httpClient.GetJsonAsync<Employee[]>("api/employees");

}

}